(19)日本国特許庁（ＪＰ）　(12) **公開特許公報（Ａ）**　(11)特許出願公開番号
特開2002－82926
（P2002－82926A）
(43)公開日　平成14年３月22日(2002.3.22)

(51)Int.Cl.[7]		識別記号	FI		テーマコード（参考）
Ｇ０６Ｆ	15/177	６７８	Ｇ０６Ｆ 15/177	６７８Ｈ	５Ｂ０４２
	9/06	ＺＥＣ	9/06	ＺＥＣ	５Ｂ０４５
	11/36		11/28	３４０Ａ	５Ｂ０７６
	11/28	３４０	11/34	Ｓ	
	11/34		9/06	６２０Ｒ	

審査請求　未請求　請求項の数18　ＯＬ　（全 15 頁）

(21)出願番号　特願2000－270341(P2000－270341)

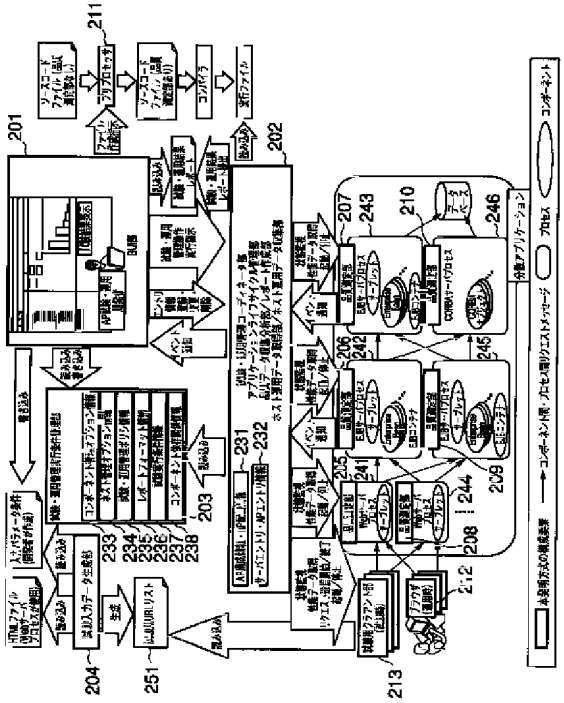
(22)出願日　平成12年９月６日(2000.9.6)

(71)出願人　000004226
日本電信電話株式会社
東京都千代田区大手町二丁目３番１号
(72)発明者　田中　博樹
東京都千代田区大手町二丁目３番１号　日本電信電話株式会社内
(74)代理人　100058479
弁理士　鈴江　武彦　（外２名）

Ｆターム(参考)　5B042 GA12 HH11
5B045 BB49 BB50 JJ08
5B076 EC05

(54)【発明の名称】　分散アプリケーション試験・運用管理システム

(57)【要約】
【課題】　分散ＡＰの構成要素であるサーバプロセスやコンポーネント単位で信頼性、性能の評価、ボトルネックの特定をすることにより分散アプリケーション単位での試験・運用管理を行うことを可能とする。
【解決手段】　分散アプリケーションのソースコードファイル群にコンポーネントでの性能データを測定する品質測定部(313,316,319)を埋め込み、その後にそれらのソースコードファイルをコンパイラに通して、サーバプロセスを起動・動作させるために必要なサーバ実行ファイルを作成して動作させ、品質データ収集分析部(306)がその品質測定部から個々のコンポーネントの性能データを収集し(382-384)、またアプリケーションライフサイクル管理部(305)から正常動作性のデータを収集する(392)。



【特許請求の範囲】
【請求項１】　ソフトウェア部品であるコンポーネント間あるいはそのコンポーネントを内包するプロセス間の通信を実行するコンポーネント間通信プラットフォームあるいはプロセス間通信プラットフォームを一種類以上含む分散アプリケーション動作環境を用い、
該システムは：コンポーネントやそのコンポーネントを内包するプロセスなどであるサーバを起動及び停止するために必要な情報をサーバエントリとして保持し、
一つのサービスを提供する一つ以上のサーバの集合である分散アプリケーション単位での起動や停止を行うために必要なサーバエントリの集合をアプリケーションエントリとして保持し、
そのアプリケーションエントリを用いて分散アプリケーションの起動や停止を制御するとともに、アプリケーションのリアルタイムな構成情報を保持するアプリケーションライフサイクル管理部と、
サーバに他のコンポーネントあるいはプロセスから処理要求が渡されるとき及びサーバから処理結果が返されるときに、その処理の開始時刻と終了時刻、処理名、及びその処理を実行したサーバ名あるいはそのサーバがコンポーネントである場合はそのクラス名を含む処理性能データを取得し、
サーバが他のサーバに処理を要求し処理結果を受けるときに、その処理要求発行時刻と処理結果受信時刻、要求した処理の名前、及びその処理要求発行先のサーバ名あるいはそのサーバがコンポーネントである場合はそのクラス名を含む応答性能データを取得する性能データ取得部と、
該性能データ取得部が取得した処理性能データと応答性能データから、個々のサーバでの処理実行時間、処理実行頻度、処理要求応答時間などを算出し、それら各々について予め規定された閾値を超過した場合にイベントメッセージを発生させる品質測定部と、
任意に選択された一つの分散アプリケーションを構成する個々のサーバに関する処理実行時間、処理実行頻度、処理要求応答時間などのデータを、そのデータを保持する品質測定部群から収集して提供し、そのデータとアプリケーション構成情報やサーバ間の依存関係情報から分散アプリケーションの性能品質確保のボトルネックとなっている個所を特定し、
また、任意に選択された一つの分散アプリケーションを構成する個々のサーバで発生した、異常終了や異常状態への遷移などの異常事象に関する、時刻、プロセスＩＤ、コンポーネント名、コンポーネントクラス名などの情報をアプリケーションライフサイクル管理部や品質測定部から収集して提供し、分散アプリケーションの信頼性品質確保のボトルネックとなっている個所を特定する品質データ収集分析部と、を具備することを特徴とする分散アプリケーション試験・運用管理システム。

【請求項２】　請求項１記載の分散アプリケーション試験・運用管理システムにおいて、
前記品質測定部がＥＪＢ（Ｅｎｔｅｒｐｒｉｓｅ　Ｊａｖａ　Ｂｅａｎｓ）コンポーネントの処理性能データを収集するにあたり、
品質測定部の一部である性能データ取得部が、コンポーネント間通信プラットフォームあるいはプロセス間通信プラットフォームに内在し、前記品質測定部の一部がＥＪＢコンポーネントに処理要求が届く直前とＥＪＢコンポーネントから処理結果が出された直後に任意の処理を実行するインタセプタとして動作し、
ＥＪＢコンポーネントに処理要求が届く直前の時刻（処理開始時刻）、ＥＪＢコンポーネントから処理結果が出された直後の時刻（処理終了時刻）、実行した処理の名前、及びその処理を実行したＥＪＢコンポーネント名あるいはそのＥＪＢコンポーネントのクラス名を含む処理性能データを性能データ取得部が取得し、品質測定部が保持することを特徴とする、分散アプリケーション試験・運用管理システム。
【請求項３】　請求項１記載の分散アプリケーション試験・運用管理システムにおいて、
前記品質測定部がＥＪＢコンポーネントの処理性能データを収集するにあたり、
品質測定部の一部である性能データ取得部が、スケルトンに埋め込まれており、
ＥＪＢコンポーネントへ他のコンポーネントやプロセスから渡される処理要求がスケルトンにある性能データ取得部を通った時刻（処理開始時刻）、ＥＪＢコンポーネントから他コンポーネントへ返される処理結果がスケルトンにある性能データ取得部を通った時刻（処理終了時刻）、実行した処理の名前、及びその処理を実行したＥＪＢコンポーネント名あるいはそのＥＪＢコンポーネントのクラス名を含む処理性能データを性能データ取得部が取得し、品質測定部が保持することを特徴とする、分散アプリケーション試験・運用管理システム。
【請求項４】　前記分散ＡＰ動作環境を用いつつ、前記品質測定部がＥＪＢコンポーネントの処理性能データを収集するにあたり、
品質測定部の一部である性能データ取得部が、ＥＪＢコンテナに埋め込まれており、
ＥＪＢコンポーネントへ他のコンポーネントやプロセスから渡される処理要求がＥＪＢコンテナにある性能データ取得部を通った時刻（処理開始時刻）、ＥＪＢコンポーネントから他コンポーネントに返される処理結果がＥＪＢコンテナにある性能データ取得部を通った時刻（処理終了時刻）、実行した処理の名前、及びその処理を実行したＥＪＢコンポーネント名あるいはそのＥＪＢコンポーネントのクラス名を含む処理性能データを性能データ取得部が取得し、品質測定部が保持することを特徴とする請求項２記載の分散アプリケーション試験・運用管

理システム。
【請求項５】　請求項１記載の分散アプリケーション試験・運用管理システムにおいて、
前記品質測定部がＥＪＢコンポーネントの処理性能データを収集するにあたり、
品質測定部の一部である性能データ取得部が、ＥＪＢコンポーネントに埋め込まれており、
ＥＪＢコンポーネントに他のコンポーネントやプロセスから処理要求が渡された直後の時刻（処理開始時刻）、ＥＪＢコンポーネントから処理結果が返す直前の時刻（処理終了時刻）、実行した処理名、及びその処理を実行したＥＪＢコンポーネント名あるいはそのＥＪＢコンポーネントのクラス名を含む処理性能データを性能データ取得部が取得し、品質測定部が保持することを特徴とする、分散アプリケーション試験・運用管理システム。
【請求項６】　請求項１記載の分散アプリケーション試験・運用管理システムにおいて、
前記品質測定部がサーブレットの処理性能データを収集するにあたり、
品質測定部の一部である性能データ取得部が、コンポーネント間通信プラットフォームあるいはプロセス間通信プラットフォームに内在し、サーブレットに処理要求が届く前とサーブレットから処理結果が出された後に任意の処理を実行するインタセプタとして動作し、
サーブレットに処理要求が届く直前の時刻（処理開始時刻）、サーブレットから処理結果が出された直後の時刻（処理終了時刻）、実行した処理の名前、及びその処理を実行したサーブレット名あるいはそのサーブレットのクラス名を含む処理性能データを性能データ取得部が取得し、品質測定部が保持することを特徴とする、分散アプリケーション試験・運用管理システム。
【請求項７】　請求項１記載の分散アプリケーション試験・運用管理システムにおいて、
前記品質測定部がサーブレットの処理性能データを収集するにあたり、
品質測定部の一部である性能データ取得部が、サーブレットに埋め込まれており、
サーブレットに他のコンポーネントやプロセスから処理要求が渡された直後の時刻（処理開始時刻）、サーブレットから処理結果が返す直前の時刻（処理終了時刻）、実行した処理名、及びその処理を実行したサーブレット名あるいはそのサーブレットのクラス名を含む処理性能データを性能データ取得部が取得し、品質測定部が保持することを特徴とする、分散アプリケーション試験・運用管理システム。
【請求項８】　請求項１記載の分散アプリケーション試験・運用管理システムにおいて、
前記品質測定部がサーブレットの処理性能データを収集するにあたり、
品質測定部の一部である性能データ取得部が、スケルトンに埋め込まれており、
サーブレットへ他のコンポーネントやプロセスから渡される処理要求がスケルトンにある性能データ取得部を通った時刻（処理開始時刻）、サーブレットから他コンポーネントやプロセスへ返される処理結果がスケルトンにある性能データ取得部を通った時刻（処理終了時刻）、実行した処理の名前、及びその処理を実行したサーブレット名あるいはそのサーブレットのクラス名を含む処理性能データを性能データ取得部が取得し、品質測定部が保持することを特徴とする、分散アプリケーション試験・運用管理システム。
【請求項９】　請求項１記載の分散アプリケーション試験・運用管理システムにおいて、
前記システムは、前記品質測定部、該品質測定部に含まれる性能データ取得部、該性能データ取得部からのイベントメッセージを受信可能な試験・運用管理コーディネータ部を具備しており、
サーバが実行した各々の処理について性能データ取得部がその処理開始時刻と処理終了時刻を取得したときに、品質測定部がその処理の処理実行時間を算出し、
その処理実行時間が規定された時間を超過した事象が規定された時間範囲で規定された回数発生した場合に、品質測定部が試験・運用管理コーディネータ部にその旨のイベントメッセージを送出し、
そのイベントメッセージを受信した試験・運用管理コーディネータ部が予め指定された内容で対処手続きを進めることを特徴とする、分散アプリケーション試験・運用管理システム。
【請求項１０】　請求項１記載の分散アプリケーション試験・運用管理システムにおいて、
前記品質測定部がＥＪＢコンポーネントの応答性能データを収集するにあたり、
品質測定部の一部である性能データ取得部が、コンポーネント間通信プラットフォームあるいはプロセス間通信プラットフォームに内在し、ＥＪＢコンポーネントから処理要求が発行された直後とＥＪＢコンポーネントに処理結果が届く直前に任意の処理を実行するインタセプタとして動作し、
ＥＪＢコンポーネントから処理要求が発行された直後の時刻（処理要求発行時刻）、ＥＪＢコンポーネントに処理結果が届く直前の時刻（処理結果受信時刻）、実行した処理の名前、及びその処理を実行したＥＪＢコンポーネント名あるいはそのＥＪＢコンポーネントのクラス名を含む応答性能データを性能データ取得部が取得し、品質測定部が保持することを特徴とする、分散アプリケーション試験・運用管理システム。
【請求項１１】　請求項１記載の分散アプリケーション試験・運用管理システムにおいて、
前記品質測定部がＥＪＢコンポーネントの応答性能データを収集するにあたり、

品質測定部の一部である性能データ取得部が、スタブに埋め込まれており、
ＥＪＢコンポーネントが他のサーバに発行する処理要求がスタブにある性能データ取得部を通った時刻（処理要求発行時刻）、ＥＪＢコンポーネントが他のサーバから受ける処理結果がスタブにある性能データ取得部を通った時刻（処理結果受信時刻）、要求した処理の名前、及びその処理要求発行先のサーバ名あるいはそのサーバがコンポーネントである場合はそのクラス名を含む応答性能データを性能データ取得部が取得し、品質測定部が保持することを特徴とする、分散アプリケーション試験・運用管理システム。
【請求項１２】　請求項１記載の分散アプリケーション試験・運用管理システムにおいて、
前記品質測定部がＥＪＢコンポーネントの応答性能データを収集するにあたり、
品質測定部の一部である性能データ取得部が、ＥＪＢコンポーネントに埋め込まれており、
ＥＪＢコンポーネントが他のサーバに処理要求を発行する直前の時刻（処理要求発行時刻）、ＥＪＢコンポーネントが他のサーバから処理結果を受信した直後の時刻（処理結果受信時刻）、要求した処理の名前、及びその処理要求発行先のサーバ名あるいはそのサーバがコンポーネントである場合はそのクラス名を含む応答性能データを性能データ取得部が取得し、品質測定部が保持することを特徴とする、分散アプリケーション試験・運用管理システム。
【請求項１３】　請求項１記載の分散アプリケーション試験・運用管理システムにおいて、
前記品質測定部がサーブレットの応答性能データを収集するにあたり、
品質測定部の一部である性能データ取得部が、コンポーネント間通信プラットフォームあるいはプロセス間通信プラットフォームに内在し、サーブレットから処理要求が発行された後とサーブレットに処理結果が届く前に任意の処理を実行するインタセプタとして動作し、
サーブレットから処理要求が発行された直後の時刻（処理要求発行時刻）、サーブレットに処理結果が届く直前の時刻（処理結果受信時刻）、実行した処理の名前、及びその処理を実行したサーブレット名あるいはそのサーブレットのクラス名を含む応答性能データを性能データ取得部が取得し、品質測定部が保持することを特徴とする、分散アプリケーション試験・運用管理システム。
【請求項１４】　請求項１記載の分散アプリケーション試験・運用管理システムにおいて、
前記品質測定部がサーブレットの応答性能データを収集するにあたり、
品質測定部の一部である性能データ取得部が、サーブレットに埋め込まれており、
サーブレットが他のサーバに処理要求を発行する直前の時刻（処理要求発行時刻）、サーブレットが他のサーバから処理結果を受信した直後の時刻（処理結果受信時刻）、要求した処理の名前、及びその処理要求発行先のサーバ名あるいはサーバがコンポーネントである場合はそのクラス名を含む応答性能データを性能データ取得部が取得し、品質測定部が保持することを特徴とする、分散アプリケーション試験・運用管理システム。
【請求項１５】　請求項１記載の分散アプリケーション試験・運用管理システムにおいて、
前記品質測定部がサーブレットの応答性能データを収集するにあたり、
品質測定部の一部である性能データ取得部が、スタブに埋め込まれており、
サーブレットが他のサーバに発行する処理要求がスタブにある性能データ取得部を通った時刻（処理要求発行時刻）、サーブレットが他のサーバから受ける処理結果がスタブにある性能データ取得部を通った時刻（処理結果受信時刻）、要求した処理の名前、及びその処理要求発行先のサーバ名あるいはそのサーバがコンポーネントである場合はそのクラス名を含む応答性能データを性能データ取得部が取得し、品質測定部が保持することを特徴とする、分散アプリケーション試験・運用管理システム。
【請求項１６】　請求項１記載の分散アプリケーション試験・運用管理システムにおいて、
前記システムは、前記品質測定部、該品質測定部に含まれる性能データ取得部、該性能データ取得部からのイベントメッセージを受信可能な試験・運用管理コーディネータ部を具備しており、
サーバが他のサーバに発行した各々の処理要求について、性能データ取得部がその処理要求発行時刻と処理結果受信時刻を取得したときに、品質測定部がその処理要求の応答時間を算出し、
その応答時間が規定された時間を超過した事象が規定された時間範囲で規定された回数発生した場合に、品質管理部が試験・運用管理コーディネータ部にその旨のイベントメッセージを送出し、
そのイベントメッセージを受信した試験・運用管理コーディネータ部が予め指定された内容で対処手続きを進めることを特徴とする、分散アプリケーション試験・運用管理システム。
【請求項１７】　請求項１乃至１６のいずれか１項に記載の分散アプリケーション試験・運用管理システムにおいて、
Ｗｅｂサーバがブラウザに返すＨＴＭＬファイルの内容からＷｅｂサーバでのサーブレット起動を要求するＨＴＴＰリクエストに対応するＵＲＬ記述部分をリストアップする試験入力データ生成部と、
その結果得られたＵＲＬリストと試験実行者が与えた引数データを用いて試験ＨＴＴＰリクエストを発生させる

試験用クライアント部を用いることを特徴とする、分散アプリケーション試験・運用管理システム。
【請求項１８】　請求項１乃至１７のいずれか１項に記載の分散アプリケーション試験・運用管理システムにおいて、
任意の２つサーバの間で、特定のアプリケーションに依存する（依存元と依存先のサーバが双方とも動作状態にあり、特定のアプリケーションを起動している場合にのみ依存関係が有効）形、あるいは特定のＡＰに依存しない（依存元と依存先のサーバが双方とも動作状態にある場合に依存関係が有効）形で、依存関係を設定し、依存関係が設定された場合には、依存先のサーバの終了が検出されたときに、依存元のサーバも終了させる試験・運用用管理コーディネータ部を具備することを特徴とする、分散アプリケーション試験・運用管理システム。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】本発明は、電子商取引サービスや情報配信サービスなどの各種情報通信サービス(以下、サービスと記す)の分野において、サーブレット〔Java Servlet API specification, SUN Microsystems Inc., http://java.sun.com/products/servlet/index.html〕、CORBA (Common Object Request Broker Architecture)〔Object Management Group, "The Common Request Broker: Architecture and Specification", Rev.2.3, ftp://omg.org/pub/docs/formal/99-12-01.pdf〕、EJB (Enterprise Java Beans)〔Enterprise JavaBeans Specification Version 1.1, Sun Microsystems, http://java.sun.com/products/ejb/index.html〕に代表される分散オブジェクト技術やコンポーネント技術(以下、CORBAオブジェクトに代表される分散オブジェクトと、サーブレットやEJBに代表されるコンポーネントをまとめて、「コンポーネント」と記す)や、コンポーネントあるいはそのコンポーネントを内包するプロセスとの間の通信を実行するコンポーネント間通信プラットフォームあるいはプロセス間通信プラットフォーム技術に関わり、特に、分散アプリケーション(以下、「ＡＰ」と記す)の試験時や運用時にＡＰソフトウェアを構成する個々のコンポーネントあるいはサーバプロセス(以下、コンポーネントとサーバプロセスをまとめてサーバと記す)の正常動作性といった信頼性や、処理実行時間、処理実行頻度、処理応答時間といった性能に関するデータの測定・収集とその収集データに基づいたＡＰ全体の信頼性や性能の評価・検証を行い、ＡＰの信頼性や性能(以下、信頼性と性能をまとめて品質と記す)のボトルネックを特定し、また、運用時に適切なＡＰ品質を確保するために運用対処を行うＡＰ試験・運用管理技術に関する。
【０００２】
【従来の技術】近年、インターネット上での電子商取引サービスやイントラネットでの社内システムなどの分野を中心に、タイムリーなサービスアプリケーションソフトウェア開発やユーザニーズの変化への迅速な対応の必要性から、コンポーネントソフトウェアを組み上げてアプリケーション(以下 単に「ＡＰ」と記す)を構築するＡＰ開発手法を支援し、さらに開発したＡＰソフトウェアの運用環境への容易な導入を支援するツールへの要望が高まっている。これに対応すべく、WWW(World Wide Web)、サーブレット、EJB、CORBAなどの技術が注目され、これらの技術を前提とした開発支援ツールやＡＰサーバ製品の導入が進んでいる。
【０００３】Web、サーブレット、EJB、CORBAなどを適用して構築したシステムの典型的な構成を図７に示す。CORBAでの標準の分散オブジェクト間通信プロトコルはIIOP(Intenet Inter-ORB Protocol)〔Java Servlet API specification, SUN Microsystems Inc., http://java.sun.com/products/servlet/index.html〕、であり、サーブレットやEJBの仕様を規定するJ2EE(Java2 Enterprise Edition)での標準のコンポーネント間通信プロトコルはRMI (Remote Method Invocation)〔Remote Method Invocation, Sun Microsystems Inc., http://java.sun.com/products/rmi/index.html〕あるいはRMI over IIOPである。IIOP、RMI、RMI over IIOPでのコンポーネント全般の場合のコンポーネント間通信方式を図８に示し、ＥＪＢコンポーネント間、及びサーブレット－ＥＪＢコンポーネント間の通信の場合コンポーネント間通信方式を図９に示す。
【０００４】一般にプロセスやホストを跨るコンポーネント間通信においては、一方のコンポーネントから送出された処理要求は、スタブ、ORB、スケルトンを介してもう一方のコンポーネントに送られ、処理結果は逆にスケルトン、ORB、スタブを介して処理要求発行元のコンポーネントに返される。このスタブ、ORB (Object Request Broker)、スケルトンを介した通信は静的起動通信方式と呼ばれる。CORBA では他に動的起動と呼ばれる通信方式があるが、開発効率や性能の点から、一般に静的通信方式の方が多く用いられる。
【０００５】また、EJBコンポーネントに処理要求が送られるときは、スケルトンに加えてEJBコンテナを介する。
【０００６】図７にあるように、アプリケーションは、電子商取引サービスなどの一つのサービスを提供するサーバ群（111-118、141-149)で構成される。分散配備されたＡＰの個々の構成要素であるの個々のサーバが他のサーバから処理要求を受け、処理を実行した結果として、サービスユーザに対してサービスが提供される。典型的なＡＰは、サービスユーザの用いるWebブラウザ(101-105)からHTTPプロトコルを介したサービス要求(以下、HTTPリクエストという)(121-125)を受け必要に応じてユーザ認証処理を行うWebサーバプロセス(111,115)、

EJBサーバプロセス(112,113,116)、CORBAサーバプロセス(113)、データベース(114,118)、Webサーバプロセスあるいは EJBサーバプロセス内で動作しWebサーバプロセスで受けたHTTPリクエストの内容に従った処理を行うサーブレット(142,146,148)、EJBサーバプロセス中で動作しビジネスロジックや各種サービスに必要な機能を実行するEJBコンポーネント(143,147,149)、及びCORBAサーバプロセス中で動作するCORBAオブジェクト(144)などを含む。Webサーバプロセスがブラウザから HTTPリクエストを受け、Webサーバプロセスあるいは EJBサーバプロセス内で起動されたサーブレットが処理を進め、さらにサーブレットとEJBコンポーネント間あるいはEJBコンポーネント間でRMI、RMI over IIOP、あるいはIIOPプロトコルを介した処理要求(以下、RMI/IIOPリクエスト)を発行しあい(131-136)処理を進めることで、結果として分散配備されたWebサーバプロセス、サーブレット、EJBコンポーネントなど全体で処理を進めてサービスユーザにサービスを提供する。

【０００７】また、現在、一般に電子商取引などを提供する事業者側のシステムやデータセンターなとで、サービスの信頼性や応答性能を確保するために、現在、システム管理として管理するホストのCPU、メモリ、ハードディスクなどのハードウェアリソースの正常性や使用率を監視することが主に行われている。また、ＡＰ管理としてグループウェア、データベースシステム、ERPなど、ソフトウェアとしてひとかたまりで提供され分散化されていないパッケージ製品の動作正常性を監視することが主に行われている。

【０００８】

【発明が解決しようとする課題】しかし、Web、サーブレット、EJB、CORBAなどの技術を適用して構築し、分散配備して運用する分散ＡＰの信頼性や性能といった品質を測定し、信頼性や処理時間あるいは応答時間が悪化したときに、そのボトルネックとなっているサーバを分散ＡＰ構成要素の中から特定することで分散ＡＰの品質を評価・検証することを支援し、さらに実装改善や運用対処で品質の保証を図ることを支援するための分散ＡＰ試験・運用管理機構は未だ十分確立されていない。

【０００９】電子商取引などの競争の激しい分野で他事業者との差別化を図るためには、サービス提供前の試験により品質の評価検証を行い、必要に応じて実装改善や運用対処などにより品質改善を図ることで品質の確保されたサービスをサービスユーザ提供することが有効である。また、サービス導入後でユーザがサービスを利用するときに所定の品質が確保されているかを測定し、必要に応じて実装改善や運用対処を行いサービス品質の確保を図ることが必要である。今後はEJBコンポーネントなど流通しているコンポーネントを組み合わせ、必要に応じて追加拡張して所望のサービスを提供するＡＰの構築を行うＡＰ開発手法が主流になると予想されるが、コンポーネントベースのＡＰで、サービス利用時にサービスユーザから見たＡＰの品質が低下したときに品質劣化の主要因となっているコンポーネントやサーバプロセスを特定する機構が未だに確立されていない。現状では、コンポーネント技術を用いてＡＰを迅速に開発できても、その品質劣化を検出したときに、ＡＰソフトウェアに内在するボトルネック個所を特定するのに未だ多くの時間と労力が費やされている。ＡＰ試験・運用中におけるサービス品質劣化時にＡＰを構成するサーバ群の品質の測定を行い、どのコンポーネントやサーバプロセスについて実装改善や運用対処を行えばよいかを判断するための正確な情報を迅速に提供する機構(システム)が望まれる。

【００１０】本発明の目的は、ＡＰを構成する個々のサーバの品質を測定しデータ収集してＡＰ(構成要素であるサーバ群全体)としての品質の評価・検証を行い、ＡＰの品質確保のボトルネックとなるサーバを迅速かつ正確に特定し、必要に応じて自動運用対処することで、ＡＰの実行を通して提供されるサービスの品質の改善と確保を支援する機構を実現することにある。

【００１１】

【課題を解決するための手段】本発明は 分散オブジェクト技術やコンポーネント技術(以下、分散オブジェクトとコンポーネントをまとめてコンポーネントと記す)に基づき、コンポーネントあるいはそのコンポーネントを内包するプロセス(以下、コンポーネントとプロセスをまとめてサーバと記す)との間の通信を実行するコンポーネント間通信プラットフォームあるいはプロセス間通信プラットフォームを用いる一つ以上のホスト(コンピュータと同義)に分散配備され、一つのサービスを提供する一つ以上のサーバの集合（本明細書中 「分散アプリケーション」と定義する）について、サーバに他のコンポーネントあるいはプロセスから処理要求が渡されるとき及びサーバから処理結果が返されるとき、その処理の開始時刻と終了時刻、処理名、及びその処理を実行したサーバ名あるいはそのサーバがコンポーネントである場合はそのクラス名を含む処理性能データを取得して保持し、サーバが他のサーバに処理を要求し処理結果を受けるときに、その処理要求発行時刻と処理結果受信時刻、要求した処理の名前、及びその処理要求発行先のサーバ名あるいはそのサーバがコンポーネントである場合はそのクラス名を含む応答性能データを取得して保持し、その取得した処理性能データと応答性能データから、個々のサーバでの処理実行時間、処理実行頻度、処理要求応答時間などを算出し、それら各々について予め規定された閾値を超過した場合にイベントメッセージを発生させる品質測定部と、任意に選択された一つの分散アプリケーションを構成する個々のサーバに関する処理実行時間、処理実行頻度、処理要求応答時間などのデータを、そのデータを保持する品質測定部群から収集して

提供し、そのデータとアプリケーション構成情報やサーバ間の依存関係情報から分散アプリケーションの性能品質確保のボトルネックとなっている個所を特定し、さらに、任意に選択された一つの分散アプリケーションを構成する個々のサーバで発生した、異常終了や異常状態への遷移などの異常事象に関する、時刻、プロセスＩＤ、コンポーネント名、コンポーネントクラス名などの情報を収集して提供し、分散アプリケーションの信頼品質確保のボトルネックとなっている個所を特定する品質データ収集分析部とを用いて、ＡＰを構成する個々のサーバの品質測定とアプリケーションとしての品質評価検証を行い、ＡＰの品質確保のボトルネックとなっているサーバを迅速かつ正確に特定する。

【００１２】また、本発明のさらに別の態様によれば、必要に応じて自動運用対処するように試験・運用管理実行条件管理及び試験・運用管理コーディネータ部を備えていることを特徴とする。

【００１３】

【発明の実施の形態】以下、本発明の実施の形態を説明する。

【００１４】図１は本発明にかかる分散ＡＰ試験・運用管理装置の各構成要素間の関係を表す図であり、図２はその運用時のプロセス構成を示す図である。本発明の実施の形態にかかる分散ＡＰ試験・運用管理装置は、以下の構成部を有する。なお 図１　２において説明の便宜上同一構成要素に異なる参照符号を附したが これは構成要素が異なることを意味するものではない。

【００１５】（１）　GUI部(図１、201、図２、301)
ＡＰ試験実行者や運用管理者からＡＰ操作要求を受け、試験結果データや運用管理状況データを表示する。

【００１６】（２）　ＡＰライフサイクル管理部　(図１、202、図２、305，307，308，311,314，317)
以下の機能を有する。

【００１７】
(2-1)ＡＰの構成情報と配備情報(図１、231)の保持
ＡＰのコンポーネント構成情報を保持する。また、それら個々のコンポーネントについて、どのホストで動作しているか(及び動作可能であるか)についての配備情報を保持する。

【００１８】(2-2)サーバエントリとＡＰエントリ情報(図１、232)の保持
サーバプロセスの起動や停止を行うために必要な情報をサーバエントリとして保持し、さらに、サーバの集合であるＡＰの起動や停止を行うために必要な情報を保持するＡＰエントリとして保持する。サーバエントリには、サーバプロセスの動作ホスト名、実行ファイルの絶対パス名、サーバプロセス起動コマンドなどが含まれる。ＡＰエントリは、サーバエントリの集合として設けられ、ＡＰの構成要素であるサーバ群の起動や停止を行うために用いられる。

【００１９】(2-2)サーバプロセスの(再)起動と停止
(2-3)サーバプロセスの動作状態監視
サーバプロセスの動作状態を監視し、起動、正常終了、異常終了、異常状態などの状態変化を検出したときにイベントメッセージを発生・送出する。プロセスの起動は、後述の品質測定部の起動時に送出されるメッセージをＡＰライフサイクル管理部が受けることで検出する。プロセスの正常終了は、オペレーテイングシステムが提供するシグナルやシステムコールなどを用いて検出する。プロセスの異常終了は、アプリケーションライフサイクル管理部からサーバプロセスへのポーリングの異常や、オペレーテイングシステムが提供するシグナルやシステムコールなどを用いて検出する。プロセスの異常状態は、サーバプロセスへのポーリングの異常から検出する。また、単位時間あたりのアプリケーション構成要素のいずれかのサーバプロセスでの異常終了や異常状態の発生の回数をアプリケーションの異常発生頻度として定期的に計測し、異常発生頻度の少なさを動作正常性として扱う。

【００２０】(2-4)ＡＰの(再)起動と停止
(2-5)試験用クライアント(図１、213、図２、310)の起動と停止
アプリケーションの品質試験を行うときに、試験・運用管理コーディネータ部(図１、202、図２、304)から試験操作の指示を受けて、試験・運用管理実行条件管理部(図１、203、図２、303)の保持する試験実行条件情報(図１、237)に従って試験用クライアント部(図１、213、図２、310)を起動し、サービスユーザの用いるブラウザ(図２、212、図331)の代わりにWebサーバプロセス(図１、241,244、図２、341)に対して試験用のリクエストを発生させる。また、試験終了後に起動した試験用クライアント部を停止終了させる。

【００２１】(３)　試験・運用管理実行条件管理部(図１、203、図２、303)
以下のデータを格納し要求者に提供する。

【００２２】
(3-1)　コンポーネント管理オプション(図１、233)
どのコンポーネントに対してポーリングを行うか、どのコンポーネントの処理実行時間や処理実行頻度などの性能に関するデータについてどの程度の時間間隔で収集するか、各々のコンポーネントの処理実行時間や処理実行頻度といった性能について許容される限界値(閾値)はどの程度か、どのコンポーネントについて処理実行時間や処理実行頻度などが予め規定された閾値を超えた場合にイベントメッセージを発生・送出させるかなどについてのデータ。

【００２３】(3-2)　ホスト管理オプション(図１、234)
どのホストについてCPU、メモリ、ハードディスクといったハードウェア資源の使用率のデータ収集を行うか、各々のホストについてそれら各々のリソース使用率の許

容される限界値(閾値)はどの程度がなどについてのデータ。
【００２４】(3-3)　試験・運用管理ポリシ(図１、235)
ＡＰ試験・運用管理時に生成され送出されたイベントメッセージを受け、そのイベントメッセージの内容に応じて自動的に望ましいＡＰ運用対処を本発明にかかる装置が自動で実行するために用いられるデータ。検出イベント種別、イベント内容に関する対象ホストと対象サーバ、及び対処手続きから構成されるポリシエレメントの集合として表される。
【００２５】検出イベント種別の例として以下が挙げられる。
【００２６】コンポーネントの起動、コンポーネントの正常終了/異常終了、コンポーネントのポーリング異常、コンポーネント性能閾値オプションの各項目における閾値超過の、指定された回数の超過、CPU使用率などのハードウエアリソースの、指定された値の超過、ポリシに従った自動対処手続きの実行の失敗。
【００２７】また、対処手続きは対処手続き要素を組み合わせて定義される。対処手続き要素の例としては以下が挙げられる。ＡＰ(再)起動、ＡＰ停止、ＡＰ閉塞/閉塞解除(一時停止/再開)、依存関係にあるコンポーネントの停止、サーバプロセス(再)起動、サーバプロセス停止、サーバ実行ファイルの指定ホストの指定デイレクトリへの配備、コンポーネント閉塞/閉塞解除(一時停止/再開)、本発明にかかる装置の構成要素へのイベント内容の通知、本発明にかかる装置の構成要素への実行した対処手続き内容及び対処手続き実行成功／失敗の通知、CPU使用率の最も低いホストの選択。
【００２８】(3-4)　レポートフォーマット(図１、236)
テキスト形式及びHTML形式に加工され格納される、処理実行時間、処理実行頻度、各種閾値、処理性能の保証率(処理実行時間が閾値以下の処理数/全処理数)、及び応答性能の保証率(応答時間が閾値以下の処理要求数/全処理要求数)などの品質測定結果データの表示フォーマットを指定するデータ。
【００２９】(3-5)　試験実行条件(図１、237)
試験実行時のWebサーバプロセスに対してHTTPリクエストを発行する試験用クライアントを動作させるホスト群、それらの各ホストで動作させる試験用クライアントの数、及び各々の試験用クライアントでのリクエスト発生時間間隔など、試験を実行するにあたっての試験環境設定条件を規定するデータ。
【００３０】
(3-6)　コンポーネント依存関係(図１、238)
コンポーネント間の処理呼び出し関係に基づき定義され、「依存元コンポーネント名－依存先コンポーネント名」の対の集合で表される。特定のＡＰに依存する形でも、ＡＰに依存しない形でも定義できる。特定のＡＰに依存する形で定義した場合には、そのＡＰが起動されている場合にのみその依存関係が適用される。2つのコンポーネントが依存関係にある場合、依存先のコンポーネントの停止が検出された場合に依存元のコンポーネントも強制停止させられる。
【００３１】（４）　試験・運用管理コーディネータ部(図１、202、図２、304)
ＡＰ試験実行者から試験実行の指示を受け、ＡＰライフサイクル管理部(図１、202、図２、308)に対して試験用クライアント部の起動を指示したり、ＡＰを構成する各々のコンポーネントから信頼性や性能のデータを収集することを品質データ収集分析部(図１、202、図２、306)に対して指示したりして、試験手続きを進める
また、ＡＰライフサイクル管理部(図１、202、図２、308)や後述の品質測定部(図１、205-210、図２、313,316,319)など本発明にかかるシステムの他の構成要素から異常発生の旨のイベントメッセージを受けたときに、上記の試験・運用管理実行条件管理部(図１、203、図２、303)の保持する試験・運用管理ポリシ情報(図１、235)を参照し、該当するポリシエレメントの内容に従った対処手続きを実行する。
【００３２】
(5)　試験入力データ生成部(図１、204、図２、302)
Webサーバプロセス(図１、241,244、図２、341)が扱うHTMLファイル群の内容からHTTPリクエスト発行に関与するURL (Unified Resource Locator)を検索・抽出し、さらに試験実行者から与えられた入力引数データを参照して試験用URLのリスト(図１、251)を生成する。
【００３３】
(6)試験用クライアント部(図１、213、図２、310)
試験入力データ生成部(図１、204、図２、302)が生成した試験用URLリスト(図１、251)を参照してWebサーバプロセス(図１、241,244、図２、341)に対してHTTPリクエストを発行する。
【００３４】
(7)品質測定部(図１、205-210、図２、313,316,319)
試験用クライアント部(図１、213，図２、310)から試験用HTTPリクエストが発行され、それに続きＡＰ実行にあたりWebサーバプロセス、サーブレット、EJBコンポーネントなどのサーバへ処理要求が渡されたとき及び処理結果が返されるとき(測定タイミングは、後述のプリプロセッサ部が品質測定用のソースコードを埋め込んだ場所に依存する)、それらのサーバ実行される個々の処理についての処理開始時刻、処理終了時刻、処理実行サーバ名、処理名(メソッド名)などの処理性能データを取得し保持する。このとき、処理終了時刻から処理開始時刻を引くことにより処理実行時間を算出する。
【００３５】さらに、単位時間あたりの実行開始した処理の数を処理実行頻度として定期的に算出する。処理実行時間や処理実行頻度が予め定められた閾値を超えた場合、イベントメッセージを生成し送出する。

【００３６】また、アプリケーション実行にあたりWebサーバプロセス、サーブレット、EJBコンポーネントなどの他のサーバへ処理要求が発行するとき及び処理結果を受けるときに(測定タイミングは後述のプリプロセッサ部が品質測定用のソースコードを埋め込んだ場所に依存する)、要求する処理についての処理要求発行時刻、処理結果受信時刻、要求処理名、処理要求先サーバ名などの応答性能データを取得・保持し、処理結果受信時刻から処理要求発行時刻を引いた値を処理応答時間として算出する。処理応答時間が予め定められた閾値を超えた場合、イベントメッセージを生成し、送出する。

【００３７】また、品質測定部(図１、205-210、図２、313,316,319)は、自身の起動時にＡＰライフサイクル管理部(図１、２０２、図２、305,307,308,311,314,317)と品質データ収集分析部(図１、202、図２、306)に対して、自身へ処理要求を送るために必要なリファレンス情報を送出する。これ以降、ＡＰライフサイクル管理部から品質測定部に状態確認のためのポーリングを行ったり、品質データ収集分析部がその品質測定部から性能データを収集したりすることが可能となる。

【００３８】

(8)品質データ収集分析部(図１、202、図２、306)

品質測定部が取得したデータから、個々のサーバでの処理実行時間(その履歴と平均)、処理実行頻度などのデータを作成する。さらに、ＡＰ構成情報とコンポーネント依存関係情報を用いてＡＰ構成要素サーバ間の処理呼び出し関係を調べ、その関係情報を用いてコンポーネントの集合、すなわちＡＰ全体としての信頼性や性能といった品質のボトルネック箇所を特定する。

【００３９】

(9)　レポート作成部(図１、202、図２、306)

品質データ収集部が収集したＡＰの信頼性や性能のデータを基にテキスト形式やHTML形式でレポートを作成する。

【００４０】(10)ホスト運用データ取得部(図１、202、図２、309,312,315,318)

CPU、メモリ、ハードディスクといったホストのリソースの使用率を定期的に調べる。それらのリソース使用率のいずれかが予め定められた閾値を超えた場合、イベントメッセージを生成し送出する。

【００４１】

(11)ホスト運用データ収集部(図１、202、図２、307)

定期的に、あるいは本発明の他構成要素からの要求に応じて、各ホストにおけるCPU、メモリ、ハードディスクといったリソースの使用率のデータをホスト運用データ取得部(図１、202、、図２309,312,315,318)から収集して提供する。

【００４２】(12)プリプロセッサ部(図１、211)

上記の品質測定部(図１、205-210、図２、313,316,319)をアプリケーション開発者の作成した各種サーバ実行ファイルに組み込む。

【００４３】以上の構成部を用いて、上記目的を実現する。

【００４４】［実施の形態の動作］ＡＰ試験・運用管理者は、以下の実施手順に従って分散ＡＰの試験・運用管理を行う。

【００４５】

(1)　サーバ実行ファイルへの品質測定部の組み込み

ＡＰ開発者あるいはＡＰ試験・運用管理者は、ＡＰ試験・運用管理を行うにあたり、まず、ＡＰ開発者が作成したソースコードファイル群や分散ＡＰプラットフォームあるいはＡＰサーバが自動生成したソースコードファイル群へ品質測定部を埋め込み、その後にそれらのソースコードファイルをコンパイラに通して、サーバプロセスを起動・動作させるために必要なサーバ実行ファイルを作成する。一例として、Java言語で作成したサーブレットやEJBコンポーネントのソースコードに品質測定部を埋め込み、コンポーネントの品質測定を可能とするためのサーバ実行ファイルを作成する手順を図３に示す。また、サーバにおける処理開始時刻あるいは処理要求発行時刻を取得するために埋め込まれるソースコードの一例を図４に、処理終了時刻あるいは処理結果受信時刻を取得するためのソースコードの一例を図５に示す。また、2つのソースコード埋め込み方法の例を図６に示す。

【００４６】この手順に従いサーブレットの処理性能を計測するための品質測定部を埋め込む場合は、(a)ＡＰ開発者の作成したサーブレットのソースコードでの各々のメソッド記述の先頭部分に処理開始時刻を取得するためのソースコードを挿入するとともに、最終部分に処理終了時刻を取得するためのソースコードを挿入する(図６中の「パターン1」の方法)か、あるいは(b)スケルトンのソースコードがＡＰ開発者の作成したコードを呼び出す直前の部分に処理開始時刻を取得するためのソースコードを挿入するとともに、直後の部分に処理終了時刻を取得するためのソースコードを挿入する(図６中の「パターン2」の方法)。その後にコンパイラを用いてサーバ実行ファイルを作成する。

【００４７】また、この手順に従いEJBコンポーネントの処理性能を計測するための品質測定部を埋め込む場合は、(c)アプリケーションサーバ製品付属のツールが作成したソースコードファイルのEJBコンテナ記述中にある、EJBコンポーネントのメソッドを呼び出す記述の直前に処理開始時刻を取得するためのソースコードを、直後に処理終了時刻を取得するためのソースコードを挿入する(図６の「パターン2」の方法)か、(d)スケルトンのソースコードがEJBコンテナあるいはEJBコンポーネントを呼び出す直前の部分に処理開始時刻を取得するためのソースコードを、直後の部分に処理終了時刻を取得するためのソースコードを挿入する(図６中の「パターン2」の方法)か、あるいは(e)ＡＰ開発者の作成したEJBコン

ポーネントのソースコードでの各々のメソッド記述の先頭部分に処理開始時刻を取得するためのソースコードを、最終部分に処理終了時刻を取得するためのソースコードを挿入する(図６中の「パターン1」の方法)。その後にコンパイラを用いてサーバ実行ファイルを作成する。

【００４８】また、この手順に従いサーブレットが要求した処理に関する応答性能を計測するための品質測定部を埋め込む場合は、(a)ＡＰ開発者の作成したサーブレットのソースコードで処理要求を起動する直前に処理要求発行時刻を取得するためのソースコードを埋め込むとともに、直後に処理結果受信時刻を取得するためのソースコードを埋め込む(図６中の「パターン2」の方法)か、あるいは(b)EJBコンポーネントが呼び出すスタブのソースコードでの各メソッド記述の先頭部分に処理開始時刻を取得するためのソースコードを、最終部分に処理終了時刻を取得するためのソースコードを挿入する(図６中の「パターン1」の方法)。その後にコンパイラを用いてサーバ実行ファイルを作成する。

【００４９】この手順に従いEJBコンポーネントが要求した処理に関する応答性能を計測するための品質測定部を埋め込む場合は、(a)ＡＰ開発者の作成したEJBコンポーネントのソースコードで処理要求を起動する直前に処理要求発行時刻を取得するためのソースコードを、直後に処理結果受信時刻を取得するためのソースコードを埋め込む(図６中の「パターン2」の方法)か、あるいは(b)EJBコンポーネントが呼び出すスタブのソースコードでの各メソッド記述の先頭部分に処理開始時刻を取得するためのソースコードを、最終部分に処理終了時刻を取得するためのソースコードを挿入する(図６中の「パターン1」の方法)。その後にコンパイラを用いてサーバ実行ファイルを作成する。

【００５０】

(2)　サーバエントリ及びＡＰエントリの設定

ＡＰ試験・運用開発者は、GUI部を介してサーバエントリとＡＰエントリをＡＰライフサイクル管理部に登録する(図２、361)。

【００５１】(3)　コンポーネント監視条件や性能データ取得条件の設定

ＡＰ試験・運用開発者は、GUI部を介してコンポーネント管理オプション情報を試験・運用管理実行条件管理部に登録する(図２、361)。

【００５２】(4)　試験・運用管理ポリシの設定

ＡＰ試験・運用開発者は、GUI部を介して試験・運用合理ポリシ情報を試験・運用管理実行条件管理部に登録する(図２、361)。

【００５３】(5)　ＡＰ起動

ＡＰ試験・運用開発者は、GUI部でＡＰエントリ名を指定し、試験・運用管理コーディネータ部を通してドメインマネージャに存在するＡＰライフサイクル管理部にＡＰ起動を要求する(図２、361)。ドメインマネージャに存在するＡＰライフサイクル管理部は、指定された名前のＡＰエントリを参照し、そのＡＰエントリを構成する個々のサーバエントリを確認し、それらのサーバエントリ毎に、サーバマネージャに存在するＡＰライフサイクル管理部にサーバプロセスの起動を要求する(図２、363、364、366、368)。サーバマネージャに存在するＡＰライフサイクル管理部は、指定されたホストで指定されたサーバ実行ファイルを用いてサーバプロセスを起動する(図２、377-379)。サーバプロセスが起動されたとき、この手続きによりＡＰを構成するサーバがすべて起動され、結果的にＡＰが起動されたことになる。

【００５４】(6)　試験用クライアント起動

ＡＰの試験を行うにあたり、試験実行者は、GUI部を介して試験・運用管理コーディネータ部に試験用クライアントの起動を要求する(図２、361)。試験・運用管理コーディネータ部は、試験・運用管理実行条件管理部から試験実行条件を入手し、その内容を基に、指定されたホストで指定された数の試験用クライアントを起動するよう、サーバマネージャに存在するＡＰライフサイクル管理部に要求する(図２、363)。起動された試験用クライアントは、試験用HTTPリクエストを指定された時間間隔でWebサーバプロセスへ送出する(図２、380)。

【００５５】(7)　性能データ取得

Webサーバプロセスが試験用クライアント(試験時)やサービスユーザが用いるブラウザ(運用時)からHTTPリクエストを受ける(図２、370,380)と、Webサーバプロセスあるいはサーバプロセスで起動されたサーブレットからEJBコンポーネントへ(図２、371)、あるいはEJBコンポーネントから他EJBコンポーネントへRMI/IIOPリクエストが送られ(図２、372)、その結果コンポーネント全体でサービス実行処理が進められる。各コンポーネントにて要求された処理が実行されるとき、品質測定部にて、各コンポーネントで実行した処理について処理開始時刻、処理終了時刻、処理名（メソッド名）、処理実行サーバ名あるいはそのサーバがコンポーネントである場合はそのクラス名を含む処理性能データが取得され(図２、373-375)、そのデータを基に処理実行時間(履歴、平均とも)や処理実行頻度などのデータが算出され蓄積される。また、各コンポーネントが他のコンポーネントに処理の実行を要求し処理結果を受けるときに、品質測定部にて、各コンポーネントから発行した処理要求について、その処理要求発行時刻と処理結果受信時刻、処理名（メソッド名）、及びその処理要求発行先のサーバ名あるいはそのサーバがコンポーネントである場合はそのクラス名を含む応答性能データが取得され（図２、３７３－３７５）、そのデータを元に応答時間（履歴、平均とも）などのデータが算出され蓄積される。

【００５６】(8)　異常発生時の自動対処

ＡＰ試験時や運用時に、プロセス異常停止やコンポーネ

ントでの処理実行時間や処理実行頻度の閾値超過などの異常がサーバマネージャ内のライフサイクル管理部（図２、３０９，３１２，３１５，３１８）によって検出されたとき、その旨のイベントメッセージが試験・運用管理コーディネータ部に送出される(プロセス異常停止の場合は図２、365,367,369、閾値超過の場合は図２、389-391)。試験・運用管理コーデイネータ部は、試験・運用管理実行条件管理部で保持されている試験・運用管理ポリシ情報を参照して、対処手続きを実行する。

【００５７】一例として、プロセス異常停止の自動対処の実行例は次のようになる。まず、サーバマネージャに存在するＡＰライフサイクル管理部がWebサーバプロセスやEJBサーバプロセスなどでのプロセス異常停止を検出する(図２、377-379)。次に、サーバマネージャに存在するＡＰライフサイクル管理部は、ドメインマネージャに存在するＡＰライフサイクル管理部を通して試験・運用管理コーデイネータ部にその旨のイベントメッセージを送出する(図２、365,367,369)。試験・運用管理実行条件管理部の保持する試験・運用管理ポリシ情報から発生したイベント(この場合は「サーバ異常終了」)に関係するポリシエレメントデータを検索する。ここで、該当するポリシエレメントデータが見つからなかった場合は何も対処手続きを実行しない。該当するポリシエレメントデータが見つかった場合には、そのポリシエレメントに記載されている内容に従って対処手続きを進行させる。

【００５８】例えば・対処手続きの内容が下記(ａ)－(ｃ)であった場合、下記の手順で手続きが実行される。

【００５９】(a)CPU使用率の最も低いホストの選択
ホスト運用データ収集部から運用中の各ホストのCPU使用率情報を収集し、最も使用率の低いホストを選択する。

【００６０】
(b)選択されたホストへのサーバ実行ファイルの配備
ＡＰ試験・運用管理者の指定したホストから、上記(a)の手続きにより選択されたホストの所定のディレクトリへサーバ実行ファイルを配備する。

【００６１】
(ｃ)選択されたホストでのサーバプロセスの起動
選択されたホストの所定のディレクトリにあるサーバ実行ファイルを用いてサーバプロセスを起動する。

【００６２】
(9)　信頼性・性能データ集計とレポート作成
ＡＰ試験・運用管理ポリシを適用してＡＰの試験・運用を行っているときに、品質データ収集部が・品質データ測定部からＡＰを構成する個々のコンポーネントの性能データを収集し(図２、385-388)、ＡＰライフサイクル管理部からＡＰを構成するサーバプロセスの異常発生頻度データを収集する(図２、392)。さらに、レポート作成部が、品質データ文集部が収集したデータを基に舳信頼性や性能のレポートを作成する。

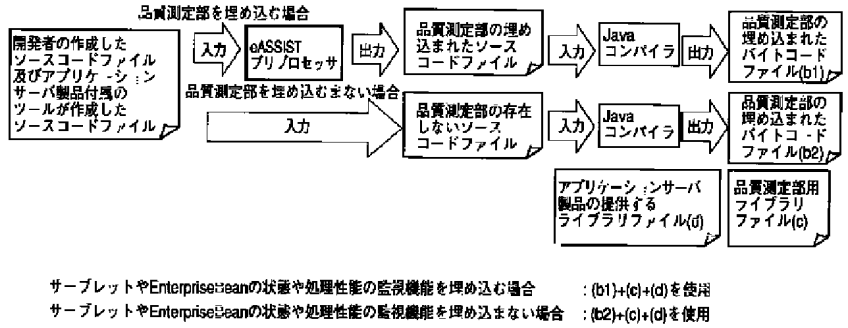
【００６３】
【発明の効果】以上説明したように、本発明によれば、サーバの集合体であるＡＰ単位での試験・運用管理を行うことが可能となる。また、分散ＡＰの構成要素であるサーバプロセスやコンポーネント単位で信頼性や性能を測定し評価することが可能になり、さらにアプリケーションの信頼性や性能を確保する上でのボトルネックとなるサーバプロセスやコンポーネントを特定することが可能となる。

【００６４】また、試験入力データ生成部や試験用クライアント部などを用いることで、試験の自動実行が可能となり、手動で行う場合に比べ試験実行の負担が軽減される。

【図面の簡単な説明】
【図１】本発明の構成要素と構成要素間の関係を説明する図である。
【図２】本発明のどの構成要素がどのプロセス上で動作するかを説明する図である。
【図３】サーバ実行ファイルへ本発明の一構成要素である品質測定部を組み込む手順の一例を説明する図である。
【図４】品質測定部の一部として処理開始時刻取得用及び処理要求発行時刻取得用に埋め込まれるソースコードの一例を示す図である。
【図５】品質測定部の一部として処理終了時刻取得用及び処理結果受信時刻取得用に埋め込まれるソースコードの一例を示す図である。
【図６】品質測定部のソースコードをサーバ実行ファイルへ埋め込む方法例を示す図である。
【図７】Web、サーブレット、EJB、CORBAなどを適用して構築したシステムの典型的な構成を示す図である。
【図８】COBRAオブジェクト、EJBコンポーネント、サーレットなどのコンポーネントが相互に通信する方式で、静的起動の場合を説明する図である。
【図９】COBRAオブジェクト、EJBコンポーネント、サーレットなどのコンポーネントが相互に通信する方式で、静的起動の場合を説明する図である。
【符号の説明】
101～105　…　ブラウザ
111,115　…　Webサーバプロセス
112,116,117　…　EJBサーバプロセス
113　…　CORBAサーバプロセス
114,118　…　データベースサーバ
121～125　…　HTTPリクエスト
131,132,134,135　…　IIOP,RMIあるいはRMI　over　IIOPを用いた処理要求
133,136　…　データベースアクセス用通信プロトコルを用いた処理要求
141,142,145,146,148　…　サーブレット

144 … CORBAオブジェクト
143,147,149 … EJBコンポーネント
151 … 分散アプリケーション
201 … GUI部
202 … 試験・運用管理コーディネータ部、アプリケーションライフサイクル管理部、品質データ収集分析部/レポート作成部、ホスト運用データ取得部/ホスト運用データ収集部
203 … 試験・運用管理実行条件管理部
204 … 試験入力データ生成部
205～210 … 品質測定部
211 … プリプロセッサ
212 … ブラウザ
213 … 試験用クライアント部
241,244 … Webサーバプロセス
242,243,245 … EJBサーバプロセス
246 … CORBAサーバプロセス
251 … 試験用URLリスト
301 …GUI部
302 … 試験入力データ生成部
303 … 試験・運用管理実行条件管理部
304 … 試験・運用管理コーディネータ部
305,308,311,314,317 … ＡＰライフサイクル管理部
306 … 品質データ収集分析部/レポート作成部
307 … ホスト運用データ収集部
309,312,315,318 … ホスト運用データ取得部
310 … 試験用クライアント部
313,316,319 … 品質測定部
331 … ブラウザ
341 … Webサーバプロセス
342,343 … EJBサーバプロセス

【図３】



サーブレットやEnterpriseBeanの状態や処理性能の監視機能を埋め込む場合 ：(b1)+(c)+(d)を使用
サーブレットやEnterpriseBeanの状態や処理性能の監視機能を埋め込まない場合 ：(b2)+(c)+(d)を使用

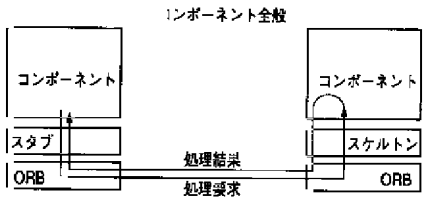
【図５】

```
finally {
recordEndTime(); //処理終了時刻を取得
}
```

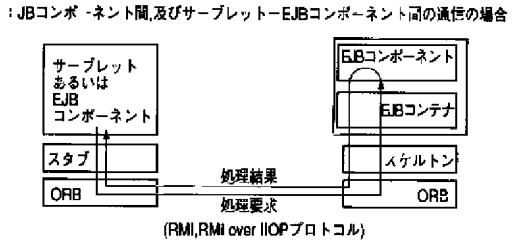
【図４】

```
recordReqInfo( "function 1" ); // 処理実行開始時刻、処理名（メソッド名）、
                               // 処理を実行したコンポーネントのクラス名を取得
                               // 引数の文字列はメソッド名（プリプロセッサが記入）
try {
```

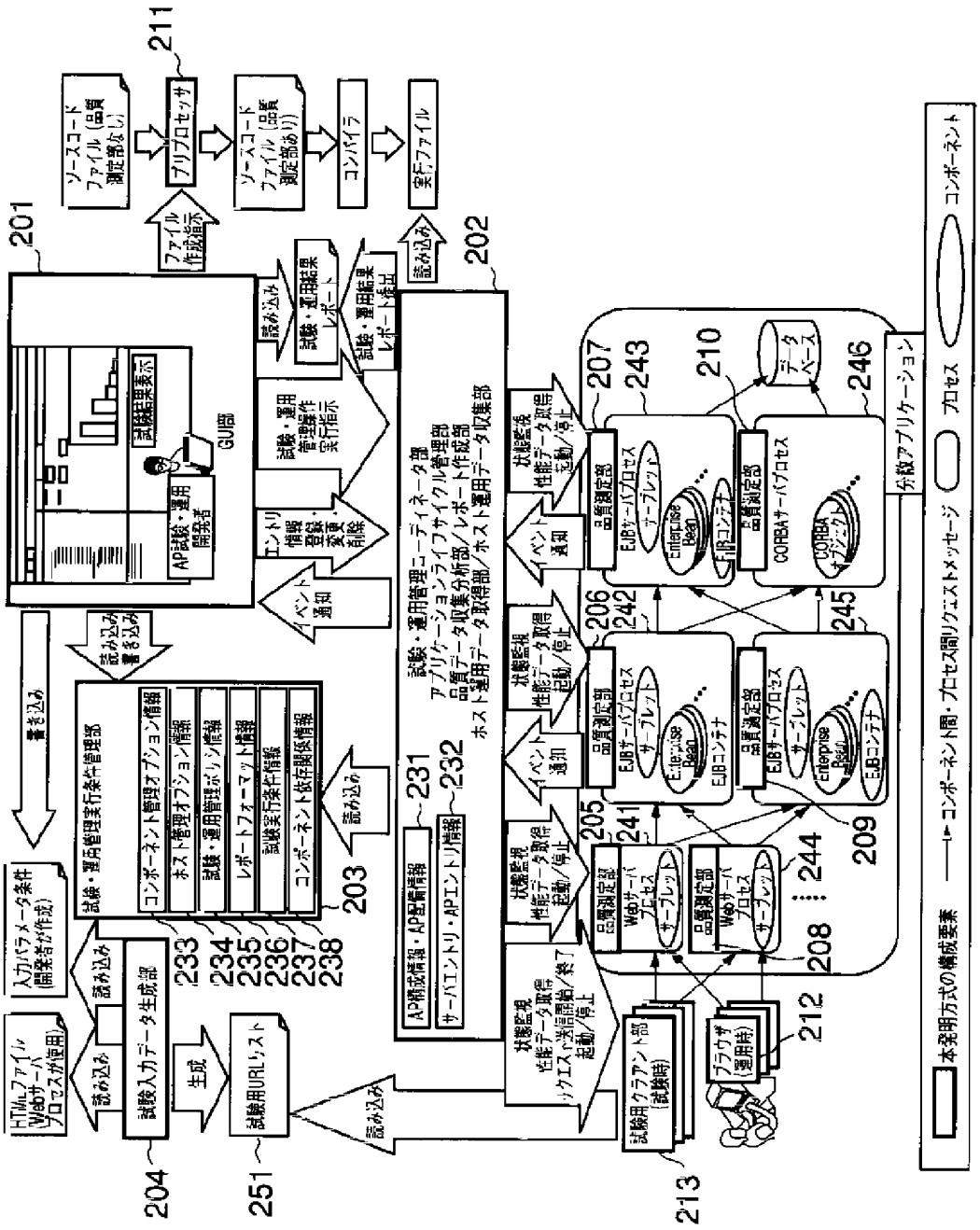
【図８】



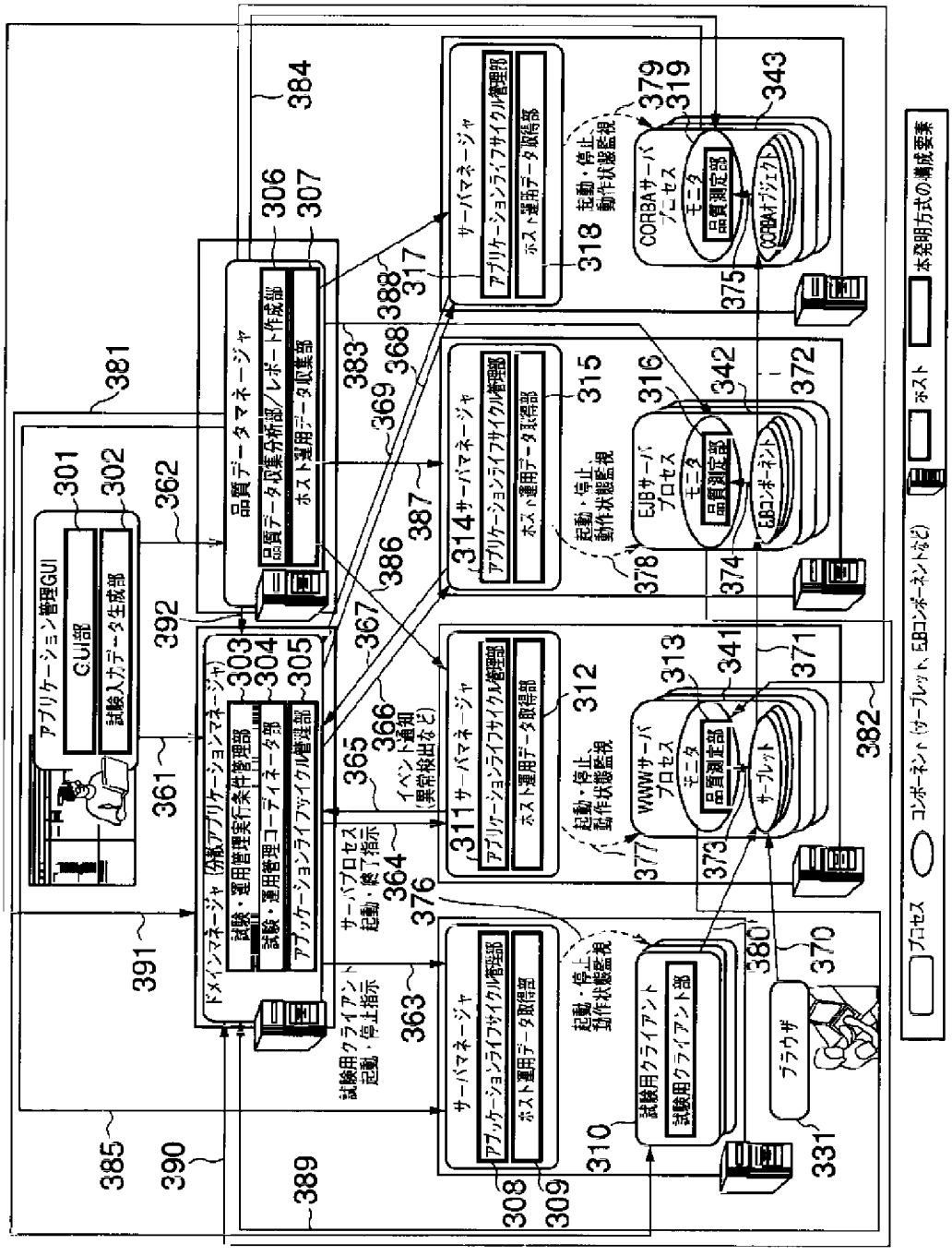
【図９】



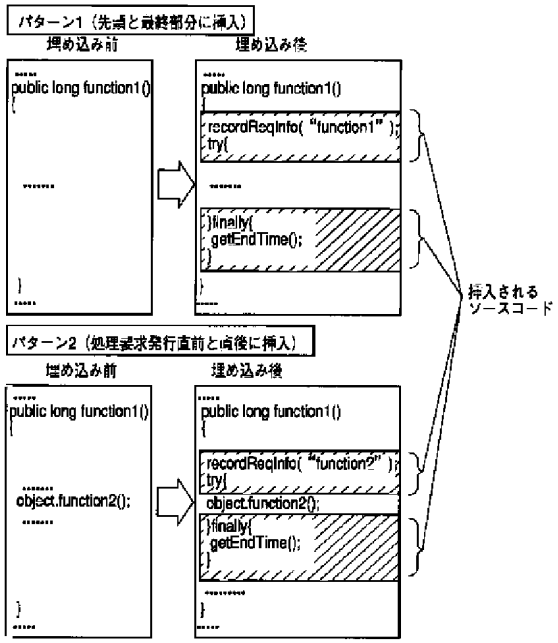
【図1】


201
211
プリプロセッサ
ソースコード ファイル（品質測定部なし）
ソースコード ファイル（品質測定部あり）
コンパイラ
実行ファイル
ファイル作成指示
読み込み
書き込み
GUI部
試験結果表示
AP試験・運用開発者
試験・運用管理実行指示
エントリ情報登録・変更・削除
イベント通知
試験・運用結果レポート
試験・運用結果レポート作成
202
試験・運用管理コーディネータ部
アプリケーションライフサイクル管理部
品質データ収集分析部／レポート作成部
ホスト運用データ取得部／ホスト運用データ収集部
203
試験・運用管理実行条件管理部
コンポーネント管理オプション情報
ホスト管理オプション情報
試験・運用管理ポリシ情報
レポートフォーマット情報
試験実行条件情報
コンポーネント依存関係情報
231
AP構成情報・AP配備情報
232
サーバエントリ・APエントリ情報
233
234
235
236
237
238
入力パラメータ条件（開発者が作成）
HTMLファイル Webサーバプロセスが使用
204
試験入力データ生成部
生成
251
試験用URLリスト
状態監視 性能データ取得起動／停止
リクエスト送信開始／終了
状態監視 性能データ取得起動／停止
イベント通知
205
206
207
241
242
243
品質測定部
Webサーバプロセス
サーブレット
EJBサーバプロセス
Enterprise Bean
EJBコンテナ
CORBAサーバプロセス
CORBAオブジェクト
分散アプリケーション
208
209
210
244
245
246
データベース
212
213
試験用クライアント部（試験時）
ブラウザ（運用時）
本発明方式の構成要素
コンポーネント間・プロセス間リクエストメッセージ
プロセス
コンポーネント

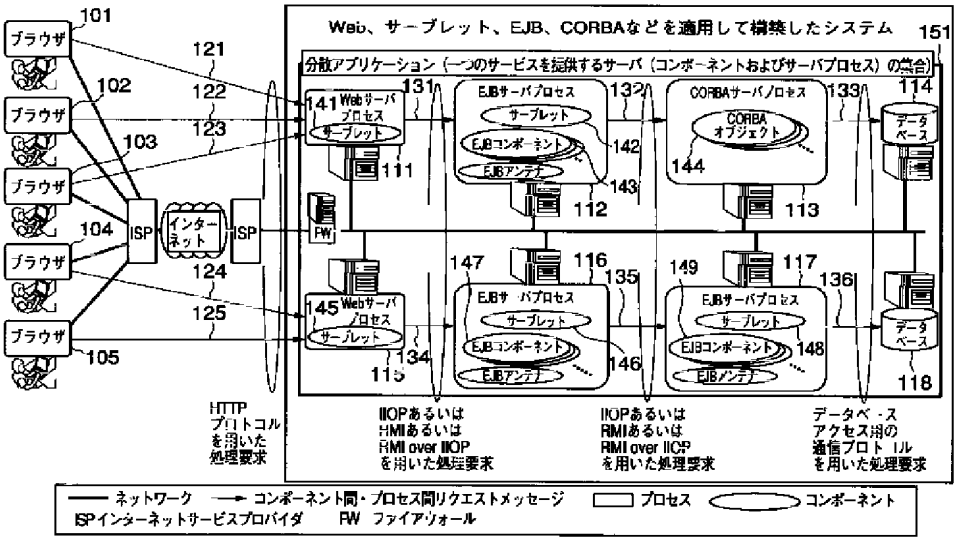
【図2】


アプリケーション管理GUI
GUI部
試験入力データ生成部
301
302
381
362
361
391
ドメインマネージャ（分散アプリケーションマネージャ）
試験・運用管理実行条件管理部
試験・運用管理コーディネータ部
アプリケーションライフサイクル管理部
303
304
305
392
品質データマネージャ
品質データ収集分析部／レポート作成部
ホスト運用データ収集部
306
307
384
383
368
369
387
386
367
365
366
イベント通知（異常検出など）
364
376
363
サーバプロセス起動・終了指示
試験用クライアント起動・停止指示
385
390
389
サーバマネージャ
アプリケーションライフサイクル管理部
ホスト運用データ取得部
308
309
310
起動・停止・動作状態監視
試験用クライアント
ブラウザ
331
370
380
311 サーバマネージャ
312
377
WWWサーバプロセスモニタ
品質測定部
373
サーブレット
313
341
371
382
314 サーバマネージャ
315
378
EJBサーバプロセスモニタ
品質測定部
EJBコンポーネント
316
342
372
374
サーバマネージャ
317
388
318
375
CORBAサーバプロセスモニタ
品質測定部
CORBAオブジェクト
379
319
343
プロセス
コンポーネント（サーブレット、EJBコンポーネントなど）
ホスト
本発明方式の構成要素

【図6】


パターン1（先頭と最終部分に挿入）
埋め込み前
埋め込み後
public long function1()
recordReqInfo("function1");
try{
}finally{
getEndTime();
挿入される
ソースコード
パターン2（処理要求発行直前と直後に挿入）
埋め込み前
埋め込み後
public long function1()
object.function2();
recordReqInfo("function2");

【図7】


Web、サーブレット、EJB、CORBAなどを適用して構築したシステム
151
分散アプリケーション（一つのサービスを提供するサーバ（コンポーネントおよびサーバプロセス）の集合）
ブラウザ
101
102
103
104
105
121
122
123
124
125
ISP
インターネット
FW
Webサーバプロセス
サーブレット
EJBサーバプロセス
EJBコンポーネント
EJBアンテナ
CORBAサーバプロセス
CORBAオブジェクト
データベース
111
112
113
114
115
116
117
118
131
132
133
134
135
136
141
142
143
144
145
146
147
148
149
HTTPプロトコルを用いた処理要求
IIOPあるいはRMIあるいはRMI over IIOPを用いた処理要求
データベースアクセス用の通信プロトコルを用いた処理要求
ネットワーク
コンポーネント間・プロセス間リクエストメッセージ
プロセス
コンポーネント
ISPインターネットサービスプロバイダ
FW ファイアウォール